UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ANA CLARA MATIVI DE SOUZA

# Evaluation of Approximate Memories using Bit Dropping in Motion Estimation for Video Coding

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engineering

Advisor: Prof. Dr. Sergio Bampi
Coadvisor: Prof. MSc. Dieison Soares Silveira

Porto Alegre
July 2019

*“The desire for knowledge shapes a man”*

— Patrick Rothfuss

# ACKNOWLEDGEMENTS

This work is dedicated to my family: Giovana, Flávio, Mariana and Matthew. It is because of your unconditional love and support that I can pursue my dreams.

To all the people that I had the pleasure to meet at Lab 215, especially Duda, Brunno, Mateus and Dieison, that made me feel very welcome when I first started. Your help was essential for my growth as a researcher.

To my dear friend Mariane, who has been on this journey with me since we started uni. I treasure your support and friendship, and I hope we can achieve many more things together.

To Professor Bampi, who made me realise how much I love Microelectronics and got me started in research. You are an amazing advisor, your support, attention and motivation got me here. Thank you for always looking out for me.

## ABSTRACT

Approximate Computing is a design paradigm that can be employed in situations where results of computation do not have to be exact – applications that can tolerate a certain amount of approximation errors and still produce useful outputs for the end-user, especially for perceptual signals processing. Approximate memories are a digital hardware design technique that brings these approximations to the data storage level. This paradigm is usually explored in Error-Tolerant Applications (ETAs), which include image and video processing, artificial neural networks, computer vision, web searches, machine learning, etc.

This work proposes the use of Approximate Memories for High Efficiency Video Coding (HEVC or H.265), with the development of a methodology to evaluate the impact of the introduction of such approximations on the quality of the compressed video and the energy consumption. The memory access patterns of the HEVC encoder are recorded as memory traces and used to extract power results. The quality degradation produced by the approximations is evaluated with objective metrics used in the video coding literature, such as Peak Signal-to-Noise Ratio (PSNR) and Bjøntegaard delta bit-rate (BD-BR).

**Keywords:** Video Coding. Motion Estimation. Memory Systems. Approximate Computing. Approximate Memory.

## RESUMO

Computação Aproximada é um paradigma de projeto que pode ser empregado em situações onde os resultados da computação não precisam ser exatos - aplicações que podem tolerar certa quantidade de erros de aproximação e ainda produzir saídas úteis para o usuário final, especialmente quando se tratar de processamento de sinais perceptuais. Memórias aproximadas são uma técnica de "design"de sistemas digitais que traz essas aproximações para o nível de armazenamento de dados. Esse paradigma geralmente é explorado em aplicações tolerantes a erros (ETAs), que incluem o processamento de imagem e de vídeo, redes neurais artificiais, visão computacional, pesquisas na Web, aprendizado de máquina etc.

Este trabalho propõe o uso de Memórias Aproximadas para Codificação de Vídeo de Alta Eficiência (HEVC ou H.265), com o desenvolvimento de uma metodologia para avaliar o impacto da introdução de tais aproximações na qualidade de saída e consumo de energia. O padrão de acesso à memória do codificador HEVC será gravado como um rastreio de memória e usado para extrair resultados de energia. A degradação da qualidade produzida pelas aproximações será avaliada com métricas objetivas utilizadas na literatura de codificação de vídeo, como a relação sinal-ruído (PSNR) e taxa de bits Bjontegaard Delta (BD-CA).

**Palavras-chave:** Video Coding. Motion Estimation. Memory Systems. Approximate Computing. Approximate Memory.

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| MH | Memory Hierarchy |
| HEVC | High Efficiency Video Coding |
| ME | Motion Estimation |
| SAD | Sum of Absolute Differences |
| TZS | Test Zonal Search |
| BER | Bit Error Rate |
| ETA | Error-Tolerant Application |
| BD | Bit Dropping |
| L1C | First Level Cache |
| LLC | Last Level Cache |
| RAM | Random Access Memory |
| DRAM | Dynamic RAM |
| SRAM | Static RAM |
| NVM | Non-Volatile Memory |
| STT-RAM | Spin-Transfer Torque RAM |
| PCM | Phase Change Memory |
| ReRAM | Resistive RAM |

# LIST OF FIGURES

# LIST OF TABLES

# CONTENTS

# 1 INTRODUCTION

The continuing advances in the IT technology industries led to more and more people having different electronic devices as a part of their daily activities. According to Cisco's forecast, there will be 3.6 networked devices per capita by 2022 (CISCO, 2017). It's important to take energy and computational constraints into account to improve the quality of products and services that are provided to the end-user of IT devices.

Digital video is constantly increasing as a media form, especially on mobile devices: video will account for about 82% of all IP traffic by 2022, up from 75% in 2017 (CISCO, 2017). Live video streaming is also on the rise, and it will account for 17% of the Internet video traffic by 2022 (CISCO, 2017). Video coding standards play a vital role in this scenario, since it is impractical and inefficient to store and transmit uncompressed videos: a 15-minute High Definition (HD) video, using the Red-Green-Blue (RGB) color space, with a resolution of 1920x1080 and at 24 frames per second takes 134 GB to store and requires a bandwidth of 149 MB/s for its transmission.

Equations 1.1 and 1.2 show the calculation of size and bit-rate for uncompressed videos. H and W refer to height and width, respectively. N is the number of bytes necessary to represent each pixel. F is the frame-rate (FPS) and t is the time duration of the video, in seconds.

$$Size = W * H * N * F * t \tag{1.1}$$

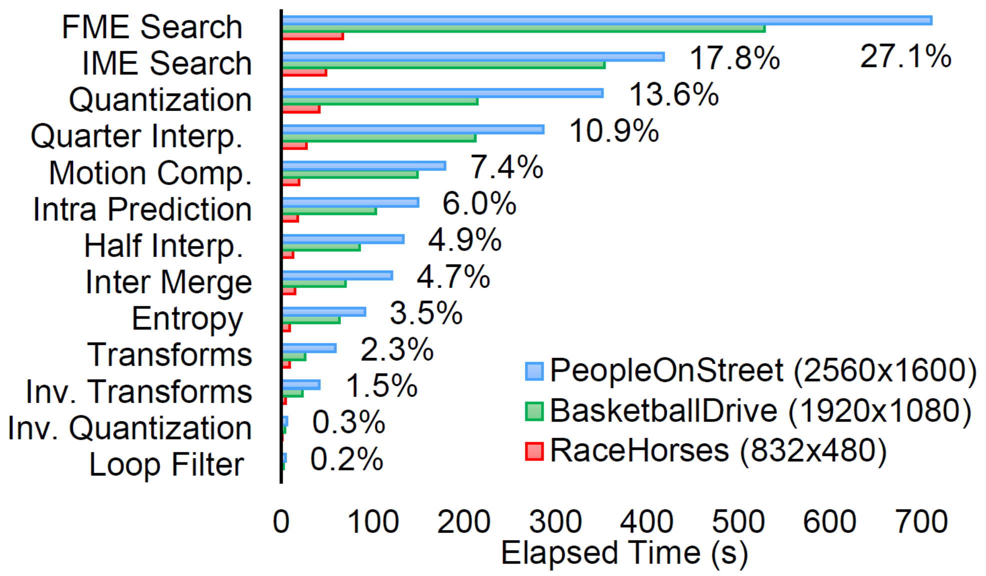
$$Bit - Rate = Size/t \tag{1.2}$$

In this scenario, it is necessary to have efficient video encoders to process videos with increasing resolutions and, at the same time, higher frame rates, keeping the bitstream as small as possible. Besides, video encoders need to be energy efficient so they can operate on battery-powered devices. Video coding takes advantage of the redundancies in and between frames to reduce the amount of data required to represent a video - which is critical when storage space and communication costs to transmit such data are considered (SHAFIQUE; HENKEL, 2014).

The need to provide efficient video encoding for ever-increasing video resolutions has led to the introduction of the HEVC (High Efficiency Video Coding, or H.265) standard in the ISO & ITU organizations in 2013 (SULLIVAN et al., 2012). The techniques adopted by HEVC use more complex data structures and considerably increase the computational effort for coding, compared to its predecessor H.264/AVC (OHM et al., 2012), (VANNE et al., 2012), (MONTEIRO et al., 2015). HEVC runs on platforms with multiple hierarchical levels of memory, and because of the large impact on runtime and power consumption associated with memory accesses, architectural improvements, encoder, and memory configurations bring interesting trade-offs between coding time, coding efficiency, and global energy consumption (SHAFIQUE; HENKEL, 2014).

HEVC was introduced to achieve better compression than its predecessor, H.264/AVC (WIEGAND et al., 2003) - about 40%-50% bit-rate savings for the same video quality (OHM et al., 2012) - and to better explore general-purpose conventional parallel processing architectures (SULLIVAN et al., 2012). HEVC delivers better quality and compression ratios, but this comes with a cost: HEVCs techniques and tools increase the computational effort of the encoding process by about 40%-70% and perform over two times more memory accesses when compared to H.264 (SHAFIQUE; HENKEL, 2014). Instead of the 16x16-sized Macroblocks defined in H.264/AVC, HEVC employs the concept of Coding Tree Unit (CTU). A CTU can be defined as a fixed block with LxL pixels, where L is typically 64.

Figure 1.1: Processing time and average time share of video encoder modules



Source: (Grellert; Bampi; Zatt, 2016)

As shown in Figure 1.1, the Motion Estimation (ME) accounts for most of the encoding time (over 50% of the encoding time, on average). There is also evidence supporting the importance of the ME on the side of memory access demands. Previous studies targeting cache memory accesses in the HEVC standard have shown that

Inter-prediction accounts for over 70% of the memory accesses performed by the encoder (MATIVI; MONTEIRO; BAMPI, 2016). By targeting the module that performs the most cache memory accesses, we can impact energy consumption with a small overall impact on the video encoder quality.

This work proposes the implementation of a technique called "Bit Dropping" in HEVC, as a way to implement Approximate Memory storage, which allows for energy and performance gains with an acceptable (possibly barely perceived) degradation in the visual quality of the encoded video. In this work, we propose a methodology that allows for the evaluation of impact in quality according to the desired level of approximation.

This work is structured as follows: In Section 2 the background concepts related to Video Coding and Memory Systems are presented, which focuses on the different interpretations of Approximate Memory and the techniques that are employed to achieve it. Related works are discussed in Section 3. In Section 4, the Methodology created for this work is presented, and in Section 5, the following Results. Finally, Section 6 presents the conclusions.

# 2 REVIEW OF VIDEO CODING AND MEMORY SYSTEMS CONCEPTS

This chapter presents the essential concepts that are used throughout this Report. First, the video coding concepts are reviewed, then the memory hierarchy principles and the corresponding memory technologies are presented. Finally, approximate memory techniques and other aspects that are covered in the related works (Section 3) are presented.

## 2.1 Video Coding

Video coding takes advantage of the spatial and temporal redundancies which are inherent to the video capturing process, in order to represent the video with a smaller amount of digital data, or a shorter bitstream (considering the transmission of the video over a serial communication channel), than otherwise a coding of every pixel in RGB, YUV or similar format would require.

### 2.1.1 High Efficiency Video Coding

The HEVC standard introduces innovations in practically all steps of the video encoding/decoding process when compared to the previous standards. Figure 2.1 presents a general block diagram of an HEVC encoder.

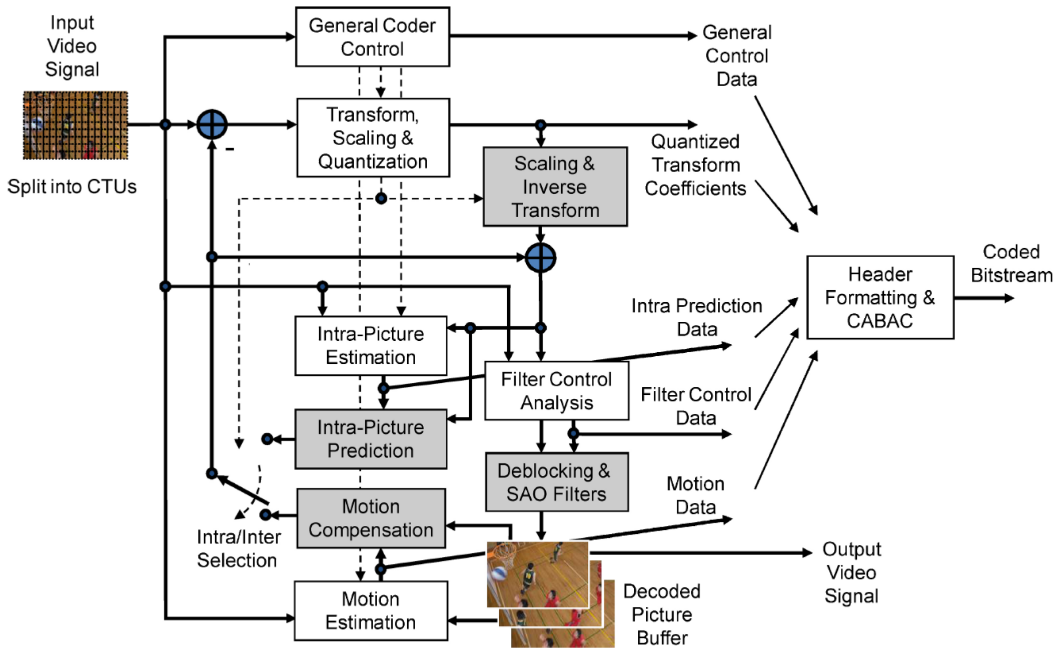
The data structures use the concept of coding tree unit (CTU) considering a size selected by the encoder, L X L pixels where L can be chosen as 16, 32 or 64. Each CTU consists of chroma and luma blocks. These blocks are divided into Coding Units (CUs), and then a decision to code with intra or inter-prediction is performed. The CUs are further divided into Prediction Units (PUs) - which become Prediction Blocks (PBs) using its chroma and luma layers that can be of sizes 4x4 to 64x64 (SULLIVAN et al., 2012). Figure 2.2 shows the breakdown of different units used by HEVC.

The encoder processes exhibited in Figure 2.1 are described as follows: (i) prediction stage; (ii) residual coding; (iii) entropy coding; (iv) decoder process and (v) filters.

In the inter-prediction step, the motion vector prediction is performed, allowing the exploration of spatial and temporal similarities between PBs.

The PBs residue is coded using cosine/sine discrete transforms, and quantization

Figure 2.1: Block diagram of the HEVC encoder.



Source: (SULLIVAN et al., 2012)

considers the distribution of frequencies after this and removes the least relevant values by a constant factor - the Quantization Parameter (QP). Large QP values mean more residual data is lost in the process. This step is optimized using Rate-Distortion Optimized Quantization (RDOQ), which consists of finding the best quantized transform coefficients choosing the ones that provide a better Rate-Distortion (RD) trade-off. The RD cost consists of calculations taking into account the distortion and the bit-stream required to represent the current CTU.

A decoding block is necessary within the encoder. This reverse process provides the current frame reconstruction. The result of this step is stored in memory and it is used as a reference picture in the next encoding frames.

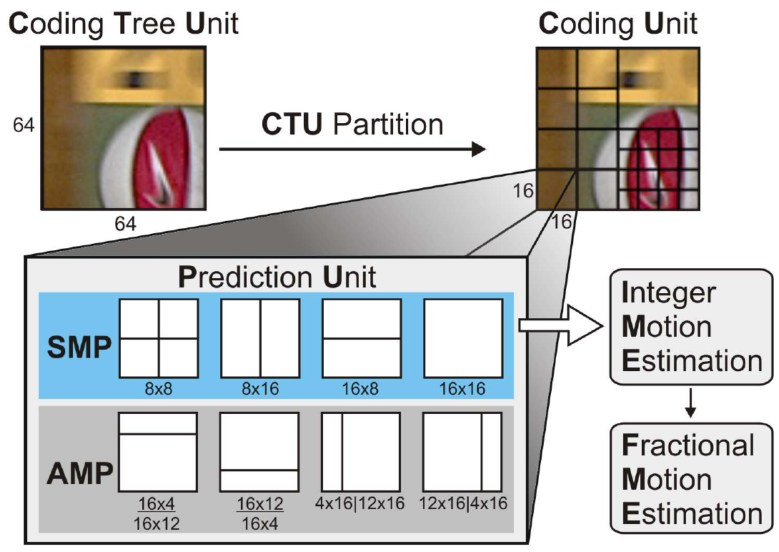
A result block is filtered by a deblocking filter within the inter-picture prediction loop. This step is operated to remove visual artifacts introduced during the process.

At the end of the process, the lossless entropy coding is responsible for generating the final bit-stream through the Context Adaptive Binary Arithmetic Coding (CABAC).

### 2.1.2 Motion Estimation

The Motion Estimation (ME) is a part of inter prediction in video encoders that is responsible for exploiting the temporal redundancies between frames. ME is applied to

Figure 2.2: Coding Units used by HEVC



Source: (Silveira et al., 2017)

each block in a frame and finds the most similar block. With that, the encoder is capable of representing the current block as the difference between the current block and its best match. This reduces the amount of information needed to represent a block, allowing for better compression.

The Sum of Absolute Differences (SAD) is a metric used by the encoder in order to measure how similar the blocks are. Each time the operation is performed, it calculates the difference between the blocks' pixel values, gets the absolute value and adds the values referring to all of the pixels. This metric is extensively applied during the Integer Motion Estimation (IME) in order to assess whether a candidate block is a better match than the current best match (Abreu et al., 2017).

## 2.2 Metrics

The metrics used for comparison of results in video coding evaluate image quality and compression. Quality metrics can be objective or subjective. Objective metrics evaluate quality measuring the distortion in the images, whereas subjective metrics consider human observers. This work does not use subjective measurements, so they will not be covered.

For objective measurements, the Peak Signal-to-Noise Ratio (PSNR) is used. PSNR is represented in decibels (dB) and is inversely proportional to the Mean Squared Error.

This metric is obtained by comparing the original and encoded video - the higher, the better. A high PSNR means that the encoding process didn't introduce a lot of distortion. Usually, when more aggressive encoding is applied (resulting in a smaller bitstream), the PSNR is lower.

The compression efficiency is evaluated by comparing bitstream sizes between original and encoded video, or between the standard and a suggested technique or approach. A smaller bitstream means a better compression. We can also look at the bitrate - the number of bits per second that must be transmitted to achieve the target frame rate.

Finally, since both measurements are valuable, we also use a metric that combines quality and compression measurements in one: Bjøntegaard Difference (BD-Rate) (Bjontegaard, 2001), which can be interpreted as the bitrate variation between the tested and reference videos, for the same quality. A positive BD-Rate means the compression is worse (larger bitrate) to achieve the same quality as the reference. The values used by BD-Rate are four pairs of PSNR, bitrate in different data points, encoded using four QP values: 22, 27, 32, 37. These pairs are used to interpolate a curve, which can be compared to the reference video curve to determine the BD-Rate (Monteiro et al., 2014).

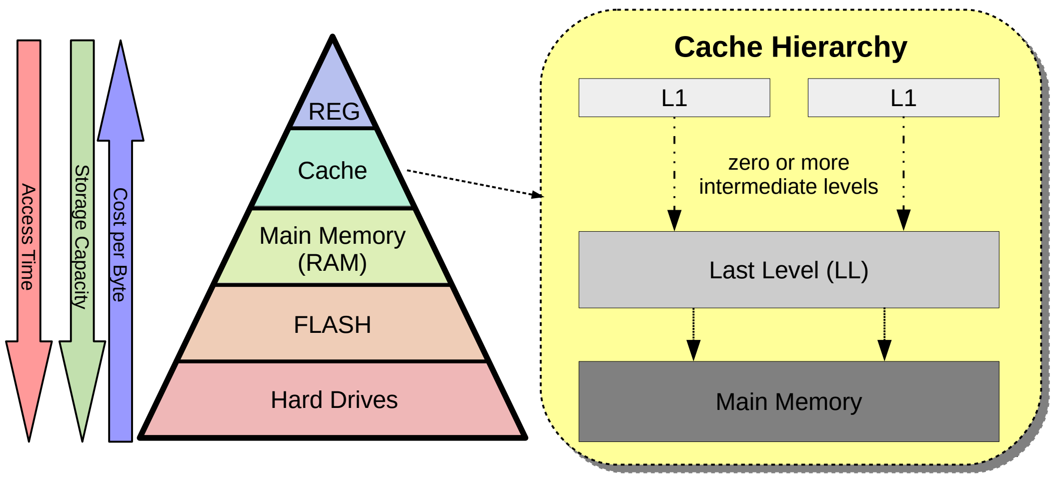
## 2.3 Memory Systems

### 2.3.1 Memory Hierarchy

The Memory Hierarchy organizes computer storage into a hierarchy based on access time. Each lower level usually has a larger capacity for storing data but is also slower to access. Figure 2.3 presents a simplified view of this organization, highlighting the cache hierarchy which follows the same principle regarding access time and storage capacity.

The Memory Hierarchy relies on the effective access of data to bridge the gap between the performance of processors and external memory (HENNESSY, 2012). Considering how data is accessed, in different layers of the hierarchy, especially in the caches, spatial and temporal localities are explored (Mittal; Vetter; Li, 2015). Temporal locality refers to the reuse of data shortly after it's first accessed. Spatial locality refers to data that is stored physically close (i.e. in the same word) since when the LL cache requests a data element from the Main Memory, a full cache line is read.

Figure 2.3: Memory Hierarchy



Source: The Author. Based on (HAO, 2014)

### 2.3.2 Random-Access Memory

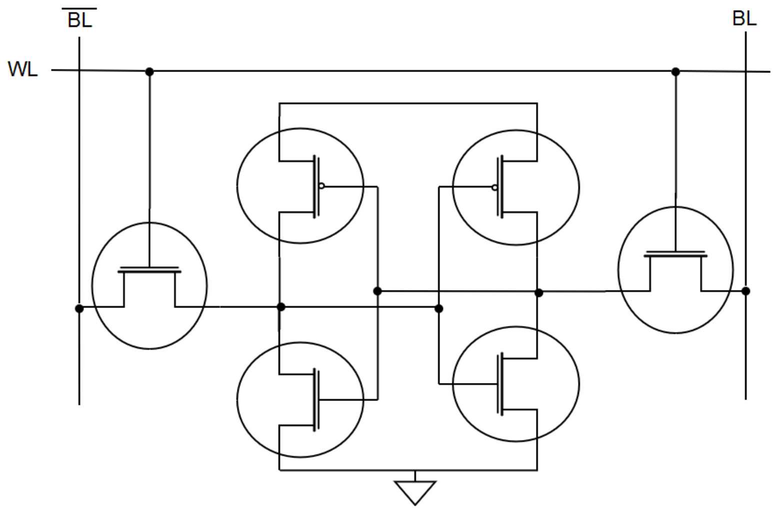
Random-Access Memory (RAM) allows for the direct access of any word in memory with identical time, regardless of where the word is located (HAO, 2014). A RAM access is performed using a bit line (BL) and word line (WL) to select the cells. Usually, multiple bits are accessed by the same address (depending on word size), and most high-capacity RAM devices or embedded-RAM blocks do access multiple words (or one entire row or block ) in one or a few cycles (JACOB, 2008).

### 2.3.3 Volatile Memories

#### *2.3.3.1 Static RAM*

Static Random-Access Memory (SRAM) has multiple cell models (JACOB, 2008). One of the most popular structures consists of 2 inverters and 2 access transistors, for a total of 6 transistors, as can be seen in Figure 2.4. Other SRAM schematics include 8 transistor cells (more robust, lower power consumption) and 4 transistor cells (higher static power consumption).

Figure 2.4: 6 Transistor SRAM Cell

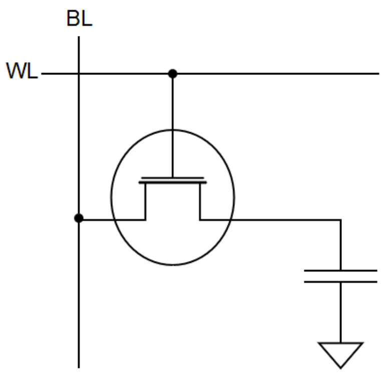


Source: (JACOB, 2008)

### *2.3.3.2 Dynamic RAM*

Dynamic Random-Access Memory (DRAM) stores bits in cells that consist of a capacitor and a transistor (JACOB, 2008). The capacitor can either be charged or discharged, representing bit states '0' or '1'. The DRAM cell is represented in Figure 2.5.

Figure 2.5: DRAM Cell



Source: (JACOB, 2008)

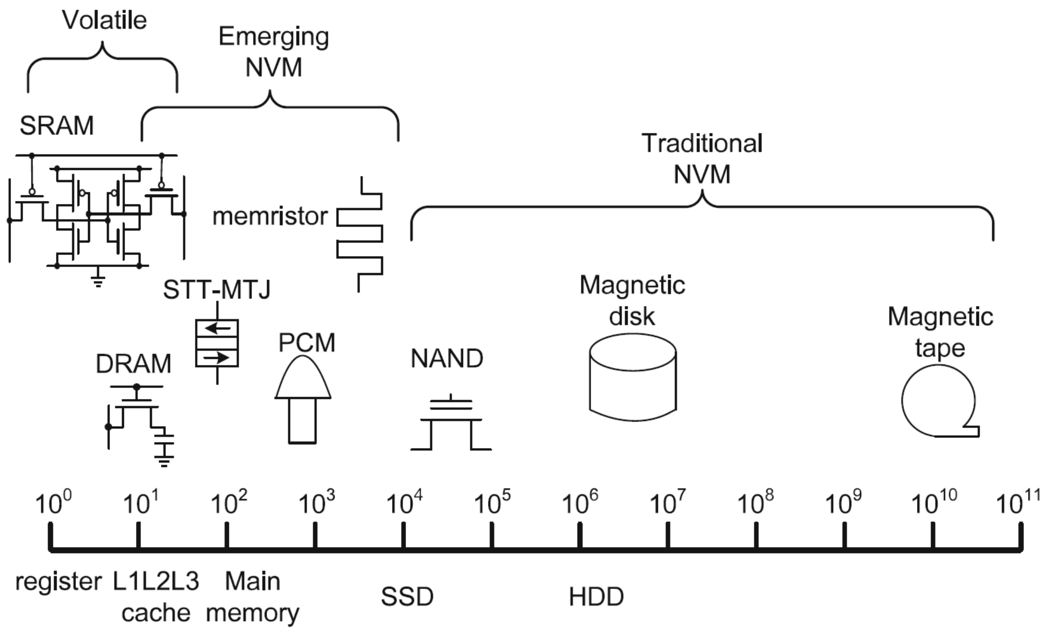
The capacitors holding data slowly discharge, so it's necessary to refresh the data stored in regular time intervals to prevent the loss of information (JACOB, 2008). The need to refresh the data in memory is the reason why it is called "dynamic", in contrast to SRAM which keeps the data as long as power is supplied. The simple cell circuitry allows DRAM to reach very high chip densities, making DRAM much cheaper per bit than SRAM.

### 2.3.4 Non-Volatile Memories

Non-Volatile Memories (NVMs) are promising technologies, that are being investigated because they are scalable and have latency and bandwidth characteristics close to DRAM, with low static power dissipation when compared to SRAMs (HAO, 2014). One of the main problems with this type of memory is the costly write operation, which exceeds DRAM and SRAM in both time and energy consumption. However, in applications with a lot more reads than writes, this type of technology can perform better than volatile memories.

Figure 2.6 presents the technologies available for memories, ordered by the number of cycles it takes to get the information. The emerging NVM technologies are much faster than the traditional ones, and due to their advantages in comparison to volatile memories, are suitable for certain applications such as video coding.

Figure 2.6: Memory Hierarchy access cycles, including NVMs

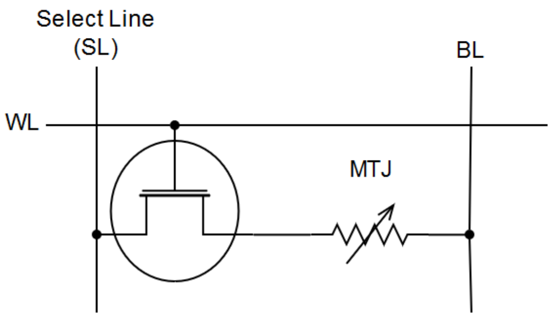


Source: (HAO, 2014)

#### 2.3.4.1 Spin-Transfer Torque RAM

Spin-Transfer Torque Random-Access Memory (STT-RAM) uses a Magnetic Tunnel Junction (MTJ) device as the storage element in memory cells (Figure 2.7) (HAO, 2014). The two possible states are defined according to the resistance (parallel/anti-parallel).

Figure 2.7: STT-RAM Cell



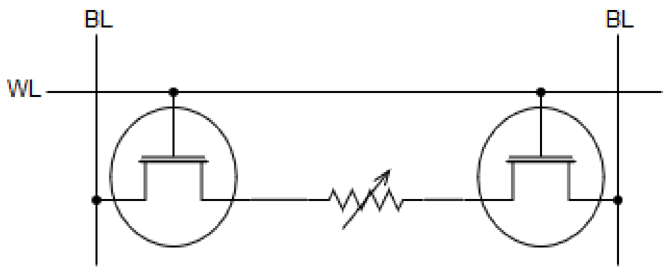
Source: The Author

This memory technology is non-volatile, presents low leakage and high-speed access (ZAND et al., 2016), which makes it a good candidate to substitute DRAMs (Suock Chung et al., 2010). The drawbacks presented by STT-RAMs are the high write current, and the wrong write cycle (2x more than the necessary for SRAMs (CHUN et al., 2013)). Finding ways to overcome these challenges has been a topic of intense research.

### *2.3.4.2 Phase Change Memory*

Phase Change Memory (PCM) presents memory cells that switch between amorphous state (high resistance or RESET) and crystalline state (low resistance or SET) by applying an electrical current (ARJOMAND et al., 2016). Figure 2.8 shows the representation of a PCM cell.

Figure 2.8: PCM Cell



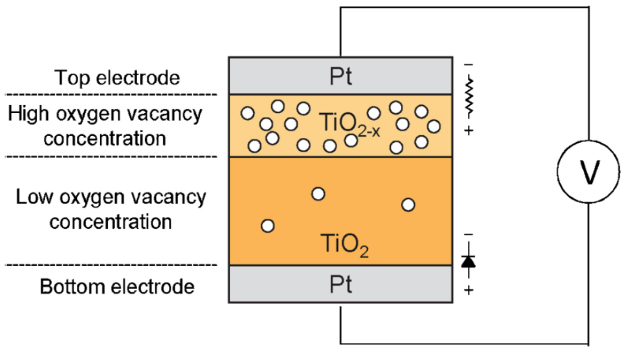
Source: The Author

In Multi-Level Cell Phase Change Memory (MLC-PCM) by dividing the resistance range it's possible to SET/RESET multiple times to represent multiple bits in a single cell. PCM presents multiple challenges that need to be addressed, such as the lifetime limitation due to the finite number of write operations a cell can endure. In MLC-PCM, a way to stretch the lifetime of the chip is to use approximations introduced by limiting the number of SET/RESET operations (thus representing different bits than intended). (TEIMOORI et al., 2018)

#### *2.3.4.3 Resistive RAM*

Resistive RAM (ReRAM), also known as memristor, is a memory based on electrically induced resistive switching effects, similar to STT-RAM and PCM (Dong et al., 2012). ReRAM cell consists of a metal oxide layer sandwiched by two metal electrodes. The electronic behavior of the metal/oxide interfaces depends on the oxygen vacancy concentration of the metal oxide layer (HAO, 2014).

Figure 2.9: ReRAM Cell


Top electrode
High oxygen vacancy concentration
Low oxygen vacancy concentration
Bottom electrode
Pt
$TiO_{2-x}$
$TiO_2$
Pt
V

Source: (Dong et al., 2012)

ReRAM prototypes have been tested, displaying promising properties on fast switching speed and low energy consumption (Dong et al., 2012).

## 2.4 Approximate Computing

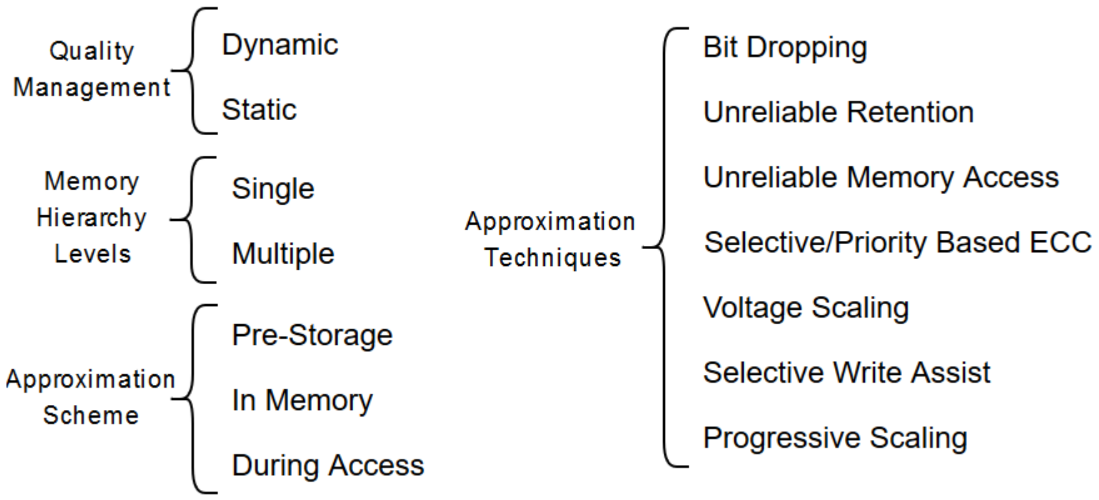
Approximate Computing is a design paradigm that can be employed in situations where results do not have to be exact – applications that can tolerate a certain amount of approximation errors and still produce useful outputs (Xu; Mytkowicz; Kim, 2016). This leads to lower power consumption and overall performance gain in computer systems. Such techniques are explored in Error-Tolerant Applications (ETAs), which include image and video processing, neural networks, computer vision, web searches, machine learning, etc.

Media applications are intrinsically inexact since the data that is encoded varies with the capture equipment used, such as the camera or sensors (Xu; Mytkowicz; Kim, 2016). Furthermore, slight changes in pixel values will not be perceived by the user due to human perceptual limitations (Xu; Mytkowicz; Kim, 2016). For other applications, it may not be feasible to get the best result due to time limitations, yet the approximate result is acceptable, like in heuristics and probabilistic algorithms (SHAFIQUE et al., 2016). It is important to note that for most systems it is mandatory to have exact storage for critical data, such as control variables, preventing errors that could cause application crashes and invalid outputs.

Approximate memory subsystems in hardware bring these approximations to the data storage level. By relaxing the output quality requirements, gains in power/performance can be achieved, which draws considerable interest from researchers, since memory is the major bottleneck for performance and power (SAMPAIO et al., 2014). The literature presents broad interpretations of the term “Approximate Memory". Paper (SAMPAIO et al., 2015) defines approximate storage as a memory unit without protection against error for reading/writing operations. Some papers, such as (FRUSTACI et al., 2016) employ techniques at the circuit level to achieve approximate storage. Other published works use software to simulate errors in memory (e.g. controlled bit-flip probability) such as (STAZI et al., 2017), (MATIVI, 2018).

In this section, we introduce several aspects of Approximate Memories that have been explored in the literature and will be further discussed in Chapter 3. Figure 2.10 summarizes the following concepts.

Figure 2.10: Approximate Memory Overview



Source: The Author

### 2.4.1 Quality Management

Management schemes proposed in the literature are usually a static solution, (i.e. a certain gain is achieved for each solution), but some dynamic management proposals can also be found. such as (TEIMOORI et al., 2018). Dynamic in this context means that it's possible to alter the quality of the output results while the application is running.

### 2.4.2 Memory Hierarchy Levels

Most papers focus on a single memory level, but it is also possible to explore approximate memory hierarchies by using multiple approximate memory levels; for example, with on-chip and off-chip memories. An exploration of this scenario is presented in (TEIMOORI et al., 2018).

### 2.4.3 Approximation Scheme

The approximation schemes refer to the moment when the approximation (loss of information) occurs. Three categories were identified:

- Pre-Storage: schemes that involve representing fewer bits per byte, a lossy compression of data which is done prior to any memory writes;
- In Memory: schemes that may introduce errors while the data is in memory, such as Unreliable Retention and Selective/Priority Based ECC;
- During Access: schemes in which the errors may occur due to inaccuracy in read or write operations, which is the case for Unreliable Memory Access and Selective Write Assist.

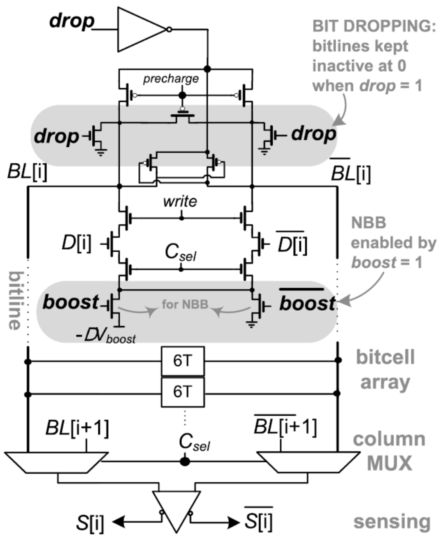
### 2.4.4 Approximation Techniques

#### *2.4.4.1 Bit Dropping*

Bit dropping refers to dropping one or more bits in each byte of the approximate memory portion. This is done by disabling the bit lines of the bits that are being approximated, as seen in figure 2.11. The usual approach is to drop bits from the Least Significant Bits (LSBs) since for figures the Most Significant Bits (MSB) influence the Peak Signal-to-Noise Ratio (PSNR) the most (ZEINALI et al., 2018), (FRUSTACI et al., 2016).

Paper (FRUSTACI et al., 2016) takes this approach one step further: since all

Figure 2.11: Bit dropping technique



Source: (FRUSTACI et al., 2016)

the bit positions incur the same energy/access cost, boosting only the MSB bits during a write, while using the LSBs as Error-Correcting Codes (ECC) for the MSB improves the robustness of the data overall while still saving energy when compared to the exact approach.

#### *2.4.4.2 Unreliable Retention*

This technique focuses on DRAMs and consists of refreshing with longer intervals than the maximum guaranteed retention of the cells, or ignoring refresh altogether. With this, the bits that store a '1' may lose their stored value due to the discharge of the capacitors. It is important to note that if data rows are accessed, this effect will be avoided for some time – in a read operation, the bits will be amplified (similar to what would happen in a refresh), and in a write operation, the data is overwritten.

#### *2.4.4.3 Unreliable Memory Access*

This refers to approximations introduced by errors during write/read operations in memory. (SAMPAIO et al., 2015) uses this concept in MLC-STT-RAMs. (STAZI et al., 2017) implements the same idea at the Operating System level - in its memory management functions - allowing for the program to allocate approximate memory, which may introduce errors in bits with a controlled probability for each memory access.

##### *2.4.4.4 Selective/Priority-Based ECC*

Selective Error-Correcting Codes (SECC) are ECCs applied to a portion of the memory cells (i.e. the bits that have more impact on quality). The other bits have no redundancy and are prone to errors (FRUSTACI et al., 2016). A Priority-Based ECC (PB-ECC) has also been proposed and validated for video coding in H.264, applying a stronger ECC for the MSBs than the LSBs (LEE et al., 2013).

##### *2.4.4.5 Voltage Scaling*

Voltage scaling (VS) is a circuit-level technique that allows the control of the supply-voltage in different domains of blocks of a system-on-chip (Gonzalez; Gordon; Horowitz, 1997). Usually, the memory blocks - like SRAM caches, embedded-DRAM, etc - use their specific supply-voltage management to control read/write margins, static power, and access times. Scaling the RAM voltage down impacts PSNR drastically to achieve better power savings. To avoid this problem and still achieve considerable power savings, the literature suggests applying voltage scaling together with other techniques.

##### *2.4.4.6 Selective Write Assist*

The Write Assist (WA) techniques aid the bit cell in changing the state during write operations and are widely used in low power SRAMs (CHANDRA; SHARMA, 2012). Some WA implementations available are word line boosting, negative bit line, VDD lowering and VSS raising. Negative Bitline Boosting (NBB) is used alongside Voltage Scaling in (FRUSTACI et al., 2014) for the MSBs only, reducing the cost added by the WA and increasing the robustness of the MSBs.

##### *2.4.4.7 Progressive Scaling*

This technique uses STT-RAM cell scaling in order to provide lower Bit Error Rate (BER) to the MSB, while LSBs are stored in cells with higher BER, following the same idea used for bit dropping which relates the quality of the output with different bit positions.

# 3 RELATED WORK

Previous works have been published to analyze the performance of video encoders for different design targets, such as energy consumption and total memory accesses. In (SINANGIL et al., 2012) a comparative analysis between coding efficiency and memory costs for HEVC is presented. Their results indicate that a higher bandwidth often results in higher power consumption, as expected. The paper (IRANPOUR; KUCHCINSKI, 2006) investigates the impact of different memory architectures on performance and energy consumption, using MPEG-4 and H.264/AVC encoders. The authors provide a comparison of dedicated memories and multilevel hierarchies, concluding that the performance improvement of dedicated memories is not significant for the target application, presenting higher energy requirements with small performance gains. Most of these works approach the Motion Estimation (ME) encoder step as the main target, as it represents a significant cost in the video coding process. Furthermore, paper (ZATT et al., 2011) reports that 90% of the energy consumption by the ME comes from on-chip storage and off-chip memory accesses.

Many papers design approximate hardware accelerators for video coding. Recently, more and more papers have been published using approximate memories for this application. Table 3 presents an overview of the literature in approximate memories according to the parameters explained in Chapter 2.

Table 3.1: Approximate memory related works

| Paper | Quality Management | Approximation Scheme | Techniques |
|---|---|---|---|
| (FRUSTACI et al., 2016) | Dynamic | Pre/During Storage | BD, VS, SECC, SWA |
| (SAMPAIO et al., 2015) | Static | During Access | Unreliable Access |
| (STAZI et al., 2017) | Static | During Access | Unreliable Access |
| (ZEINALI et al., 2018) | Static | Pre-Storage | Progressive Scaling |
| (RANJAN et al., 2017) | Dynamic | Pre-Storage | Unreliable Retention |
| (NGUYEN et al., 2018) | Dynamic | During Storage | Unreliable Retention |
| (JUNG et al., 2016) | Static | During Storage | Unreliable Retention |
| (TEIMOORI et al., 2018) | Dynamic | During Access | Unreliable Access |

Paper (FRUSTACI et al., 2016) uses a 28-nm SRAM test chip to test many different techniques. They cover bit dropping, voltage scaling, selective ECC and selective write assist, as well as combinations of these techniques at the same time. In (SAMPAIO et al., 2015), an STT-RAM cache architecture is used, and the approximation is achieved by relaxing the protection that is in the memory cells - which makes them less reliable, thus introducing the approximation. The paper (STAZI et al., 2017) implements a software layer in the OS that simulates the behavior of memory with unreliable access, which means that there is a probability of the data being approximated every time it is being accessed. How realistic is this approach at the OS level concerning the real data degradation present in a given hardware platform (processor and memory subsystems) was not discussed in the paper. In (ZEINALI et al., 2018), the idea of progressive scaling is explored with STT-RAM cells. Cells are scaled in a way that the LSBs are allocated to cells with higher BER (which can be interpreted as more approximation), and the MSBs are in cells that are prone to a lower BER. Papers (NGUYEN et al., 2018) and (JUNG et al., 2016) use DRAMs, where the authors use the properties of refresh and cell retention times to save energy and keep the most important data in the cells that are less prone to errors. In (TEIMOORI et al., 2018), the authors implement a memory hierarchy using an STT-RAM scratchpad and PCM main memory. They implemented a scheme to improve the lifetime of PCM as well.

# 4 METHODOLOGY

This chapter contains the discussion of the framework that is used to evaluate bit dropping in video coding. We begin by choosing a module of the HEVC encoder to use the approximate memory on. Then, different NVMs were simulated in order to obtain energy estimations that will be used in this methodology to assess the energy costs.

## 4.1 Framework

This methodology is mostly implemented as Python scripts. In Figure 4.1 it can be observed that three main tools compose the framework:

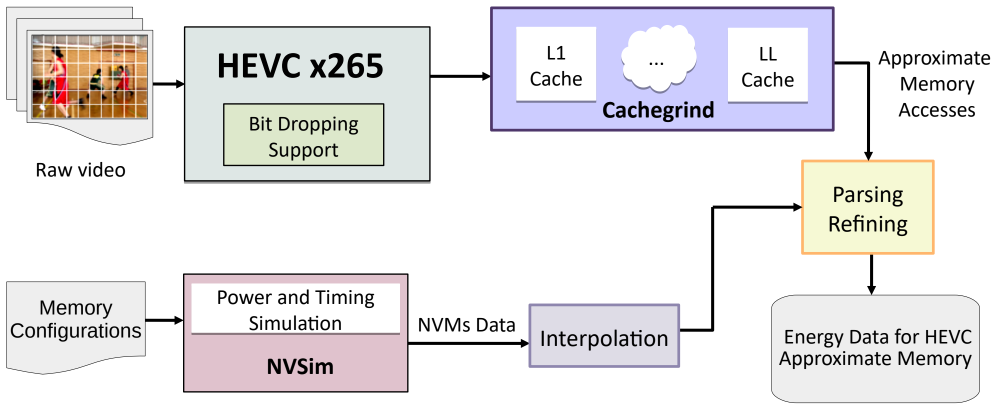
- Encode Process Data: the x265 encoder software (MULTICOREWARE, 2019) is considered as the tool to run the encoding of the test video sequences.
- Cache Memory Behavior Data: the Cachegrind tool used in this context is part of an instrumentation framework called Valgrind (NETHERCOTE; SEWARD, 2007), which allows simulating different cache memory hierarchy configurations, e.g., the number of accesses, hits, misses and read/write operations for the first level (L1C) and for the last level (LLC) of the modeled cache memory. Cachegrind is the third-party tool responsible for profiling the x265 software cache memory accesses.
- Memory Simulation Data: the power and timing data for each memory technology and memory capacity considered will be extracted from NVSim (Dong et al., 2012), which is a tool similar to Cacti that can generate accurate energy estimations for NVMs.

Note that the HEVC x265 encoder software was used in this study, but the framework could be used for other software applications. The x265 encoder uses optimizations that were developed by the coding experts (both in software and in video coding) for faster encoding. The main script was performed to target different cache memory configurations and the HEVC encoding structures can be easily adjusted by input parameters.

The Bit Dropping technique consists of disabling the Least Significant Bit Lines (LSBs) in the approximate portion of memory to reduce the dynamic energy. The dynamic energy for memory accesses is reduced for each dropped bit.

These masked bits are used in the Absolute Differences Sum (SAD) modules, for all of the reference frames. SAD is one of the most important modules used during motion

Figure 4.1: Methodology



Source: The Author

Table 4.1: Masks used for Bit Dropping

| Bit Mask | Dropped LSBs | Word sizes |
|---|---|---|
| 1111 1111 | 0 | 16, 32, 64 |
| 1111 1110 | 1 | 14, 28, 56 |
| 1111 1100 | 2 | 12, 24, 48 |
| 1111 1000 | 3 | 10, 20, 40 |
| 1111 0000 | 4 | 8, 16, 32 |

estimation in the Inter-prediction step, and by approaching the memory used for reference frames, we ensure that the approximation impact is limited to Inter-prediction and does not affect other encoder modules. The compatibility in the encoder side can be easily implemented as well.

This implementation using masks of Table 4.1 can be considered a good abstraction of the hardware implementation since, when the bit lines are deactivated in the memory, we still work with multiple bytes, each still comprising 8-bit lines. The deactivated bit lines will be read as zeros, just as the masked bits used here.

## 4.2 Experimental Setup

### 4.2.1 Video Coding Scenario

The HEVC Common Test Conditions (CTC) (BOSSEN, 2013) were considered to define the video coding setup. The sequences used in our experiments are shown in Table 4.2.1.

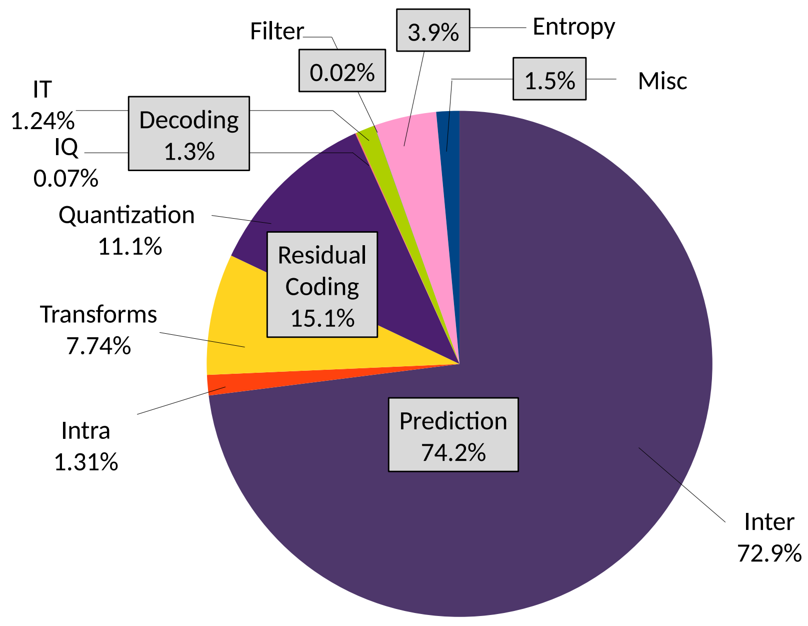
An analysis of the modules of the encoder was performed to support the decision

Table 4.2: Video Sequences

| Name | FPS | Bit Depth | Resolution |
|---|---|---|---|
| BasketballDrill | 50 | 8 | 832x480 |
| BQMall | 60 | 8 | 832x480 |
| RaceHorses | 30 | 8 | 832x480 |
| BasketballDrive | 50 | 8 | 1920x1080 |
| BQTerrace | 60 | 8 | 1920x1080 |
| Cactus | 50 | 8 | 1920x1080 |
| NebutaFestival | 60 | 10 | 2560x1600 |
| PeopleOnStreet | 30 | 8 | 2560x1600 |
| Traffic | 30 | 8 | 2560x1600 |

about which encoder parameters have the most impact on the encoder. The results of this investigation are presented in Figure 4.2. This chart shows that the most demanding module is the Inter-Prediction, with 72.9% of the total cache memory accesses, as discussed previously in Chapter 1.

Figure 4.2: Distribution of memory accesses per HEVC module



Source: The Author

### 4.2.2 Memory Configuration

#### *4.2.2.1 Cache Profiling*

For the memory access data, we used the Cachegrind configuration as seen in Table 4.2.2.1. Cachegrind can simulate only the first and last level, with any intermediate levels that could exist being abstracted. This is still a good enough abstraction since any misses in the last level of cache would incur in main memory accesses and we can model those. The sizes considered for the caches are the usual sizes for embedded cores.

Table 4.3: Cache Memory Configuration

| | |
|---|---|
| First Level Cache | 32 KB |
| Last Level Cache | 1 MB |
| Associativity | 4-way |
| Word Size | 32 bit |

In this context, results in (IRANPOUR; KUCHCINSKI, 2006) were considered, where the authors concluded that going beyond 8-way associativity provides no significant improvement. Hence, the 4-way associativity for LLC was used in further experiments.

The hardware platform used for the simulations is an Intel Core i5 with 4 cores running at 2.6 GHz and 4 GB of main memory. To ensure data cache accesses results reliability, each physical core worked with a single thread. The code was compiled for x86-64 architecture using GNU Compiler Collection (GCC) without performance optimization.

#### *4.2.2.2 Energy Estimation*

In this investigation, NVMs are used together with an approximate storage approach (implemented with bit dropping), in order to achieve more energy savings. NVMs are particularly interesting for video coding applications since inherently the memory considered will be subject to many more read accesses than write accesses - which helps to overcome some of the challenges pointed in Chapter 2.

The energy consumption estimates are provided by the NVSim tool. The first and last level caches were simulated considering word sizes of 8, 16, 32, and 64, on RRAM, SRAM, STTRAM, and PCM. The different word sizes are considered in order to have more data points to be used in interpolation, because NVSim only allows the simulation of base 2 word sizes, and the dynamic energy that will be used when bits are approximated with bit dropping needs to consider word sizes that are not base 2, as shown in table 4.1. For the first level, a 32 KB cache is simulated, and 1 MB is used for the last level. The energy results for the first level cache can be seen in Table 4.4 and the PCM SET and RESET estimations, in Table 4.5.

Table 4.4: NVM Energy Results

| Cell | Data Width | Read dynamic energy (pJ) | Read leakage energy | Write dynamic energy (pJ) | Write leakage energy | Leakage Power (mW) |
|---|---|---|---|---|---|---|
| RRAM | 8 | 1.822 | 0.0238 nJ | 5.399 | .37393 nJ | 18.540 |
| | 16 | 2.883 | 0.0177 nJ | 10.383 | .28393 nJ | 14.078 |
| | 32 | 5.012 | 0.0148 nJ | 20.352 | .23941 nJ | 11.870 |
| | 64 | 10.184 | 0.0317 nJ | 40.159 | .48471 nJ | 23.947 |
| SRAM | 8 | 6.638 | 17.599 pJ | 0.639 | 17.599 pJ | 38.999 |
| | 16 | 6.644 | 14.887 pJ | 0.685 | 14.887 pJ | 37.636 |
| | 32 | 6.658 | 13.581 pJ | 0.778 | 13.581 pJ | 36.956 |
| | 64 | 6.685 | 12.942 pJ | 0.965 | 12.942 pJ | 36.619 |
| STTRAM | 8 | 2.033 | 0.0230 nJ | 2.599 | .08763 nJ | 16.579 |
| | 16 | 3.444 | 0.0221 nJ | 4.803 | .08475 nJ | 16.033 |
| | 32 | 6.559 | 0.0227 nJ | 9.152 | .08805 nJ | 16.667 |
| | 64 | 12.438 | 0.0223 nJ | 17.844 | .08680 nJ | 16.428 |
| PCM | 8 | 1.361 | 3.7427 pJ | 1.626 | 2.8542 nJ | 14.998 |
| | 16 | 1.449 | 2.5814 pJ | 3.242 | 2.3301 nJ | 12.244 |
| | 32 | 1.612 | 1.8787 pJ | 6.483 | 2.0709 nJ | 10.884 |
| | 64 | 2.391 | 2.8771 pJ | 12.964 | 2.2308 nJ | 11.715 |

Table 4.5: PCM SET and RESET Energy

| Cell | Data Width | RESET dynamic energy (nJ) | SET dynamic energy (nJ) |
|---|---|---|---|
| PCM | 8 | 0.811 | 0.810 |
| | 16 | 1.621 | 1.621 |
| | 32 | 3.242 | 3.241 |
| | 64 | 6.482 | 6.482 |

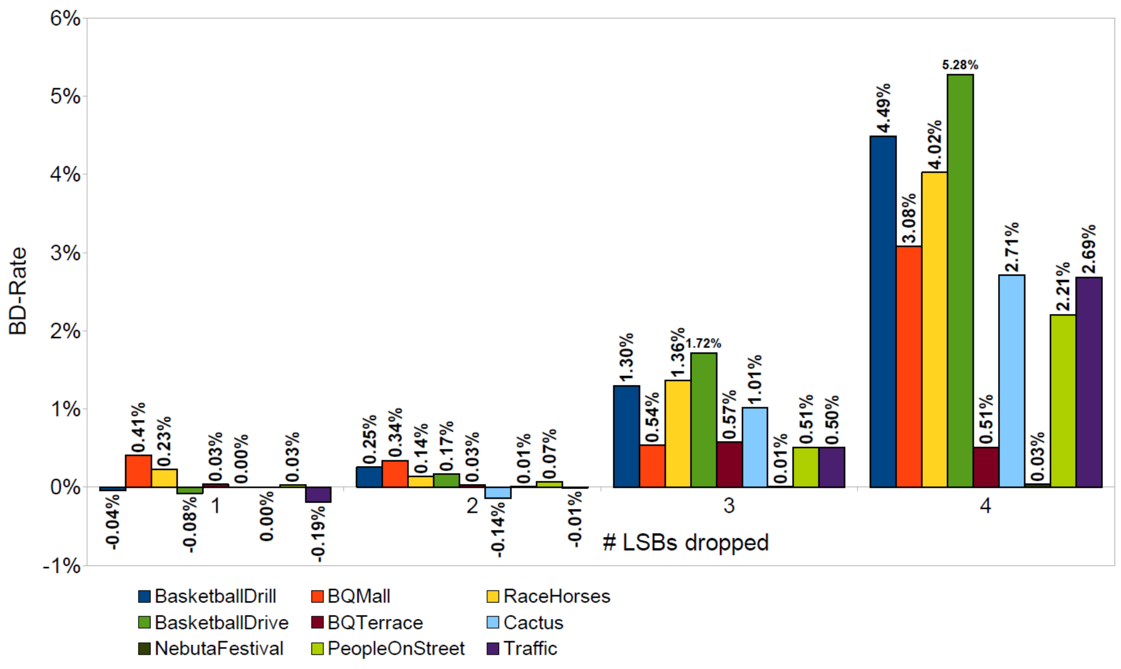
# 5 RESULTS

In this chapter, the implementation of bit dropping in x265 software will be evaluated using the objective metrics for video coding, the cache performance analysis, and energy estimation metrics.

## 5.1 BD-Rate

In order to obtain the BD-Rate, extensive simulations were done, using 9 different video sequences: Class C videos - 832x480 (BasketballDrill, BQMall, RaceHorses), Class B videos - 1920x1080 (BasketballDrive, BQTerrace, Cactus), and Class A videos - 2560x1600 (NebutaFestival, PeopleOnStreet, Traffic). 100 frames from each sequence were encoded using bit dropping and different bit masks, where the exact result (zero bits dropped) was used as the reference for every data point. Each of these simulations was performed for four different QPs in order to calculate the BD curves, as required by the quantitative BD delta methodology to compare video encoders. Figure 5.1 shows these results.

Figure 5.1: BD-Rate impact per number of LSBs dropped



The data points show that there is very little increase in bit rate when 1 and 2 LSBs are dropped for the same quality as the exact approach. In some cases, there is even an improvement in the compression quality when bit dropping is considered, which could be a result of better entropy coding ( a lossless binary arithmetic coding) at the end of the encoding process. Overall, dropping up to two bits results in a negligible bit rate increase, and could be used to leverage energy gains.

When 3 and 4 bits are dropped, we see a more noticeable impact on the BD-Rate - up to 5.3% increase in the bit rate for the same quality. This is still acceptable for many applications, especially when considering a mobile platform for video display, in which saving energy is a major concern to allow continuous viewing without battery recharging.

## 5.2 Cache Performance

In Sections 5.2 and 5.3, due to the great increase in simulation time from using Cachegrind, associated with the limited computing power at our disposal for performing this study, the simulations were limited to a Group of Pictures (GoP) and just one video sequence of each class of video resolution. Each GoP, for the encoder configurations used, consists of five frames. Table 5.2 shows the configuration used for the video encoder.

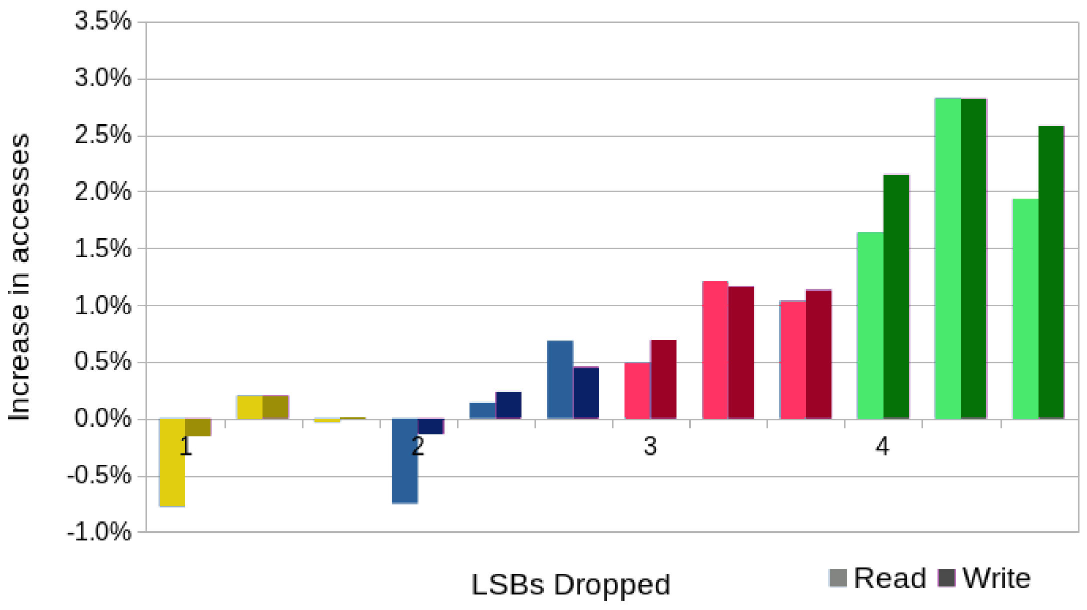
Table 5.1: Video sequences used for cache profiling

| Name | Frames | Resolution |
|---|---|---|
| BasketballDrill | 5 | 832x480 |
| Cactus | 5 | 1920x1080 |
| Traffic | 5 | 2560x1600 |

In Figure 5.2 the number of cache memory accesses is compared to the exact approach for each dropped bit. For the first sequence (BasketballDrill), the number of accesses went down when 1 and 2 bits were dropped. This might be the result of the motion estimation step stopping earlier in some of the blocks considered. In every other case, we can see an increase of less than 3% in the number of accesses. The increase is likely due to having certain blocks that take a longer time to find the best match for the motion estimation step.

Regarding the impact caused by bit dropping on the miss rates, in Table 5.2, we can see that there is very little difference in the miss rates for the first and last level - in the best case we see a reduction of 1.329% and in the worst case, an increase 1.722%, which are percentages on top of already very low miss rates. This shows that this approximation technique doesn’t incur more accesses to the main memory when compared to the exact approach. Since the number of bits dropped does not seem to impact the miss rates, the

Figure 5.2: Increase in number of cache accesses



decision of how many LSBs to drop could be based more on the energy savings and on the BD-Rate impact.

## 5.3 Energy

In order to find the data points for dropped bits, with word sizes as shown in Table 4.1, an interpolation function was written in SciLab and used to get the data points for the word sizes of 1, 2, and 3 dropped LSBs. Even though the cache profiling was done only for 32-bit words, and considering their bit dropping versions, it is better to have more data points to interpolate the energy estimation. Thus, Table 5.3 shows the interpolated data for the first level cache. The same process was done for the last level cache (LLC).

For the energy evaluation, the interpolated data offers support to simulate the bit dropping energy behavior. The deactivated bit lines, that correspond to the dropped LSBs, will result in a reduction of dynamic power, that is proportional to the number of bits dropped. Figures 5.3, 5.4, and 5.5 present the dynamic energy reduction for each sequence.

We can observe that the NVMs have the best energy reduction when dropping bits in comparison to the exact approach, being close to the ideal linear reduction. The SRAM does not scale so well in this scenario, leading us to believe that the best configuration for

Table 5.2: First and last level cache miss rates for different video sequences

| Sequence | LSBs Dropped | L1 Miss Rate (%) | LL Miss Rate (%) |
|---|---|---|---|
| BasketballDrill | 0 | 0.2030450 | 0.02271163 |
| | 1 | 0.2054960 | 0.02273629 |
| | 2 | 0.2048948 | 0.02291351 |
| | 3 | 0.2039843 | 0.02285334 |
| | 4 | 0.2032863 | 0.02277724 |
| Cactus | 0 | 0.1738354 | 0.02103040 |
| | 1 | 0.1739784 | 0.02103356 |
| | 2 | 0.1741161 | 0.02113628 |
| | 3 | 0.1737670 | 0.02083800 |
| | 4 | 0.1729797 | 0.02075086 |
| Traffic | 0 | 0.1527935 | 0.01953888 |
| | 1 | 0.1529883 | 0.01980358 |
| | 2 | 0.1525972 | 0.01943880 |
| | 3 | 0.1524330 | 0.01963800 |
| | 4 | 0.1524972 | 0.01987552 |

bit dropping would be one of the studied NVMs.

Table 5.3: Interpolated values for 32 bit word size with bit dropping

| | Data Width | Read dynamic energy (pJ) | Write dynamic energy (pJ) | RESET dynamic energy (nJ) | SET dynamic energy (nJ) |
|---|---|---|---|---|---|
| RRAM | 16 | 2.883 | 10.383 | – | – |
| | 20 | 3.408 | 12.876 | – | – |
| | 24 | 3.935 | 15.368 | – | – |
| | 28 | 4.468 | 17.861 | – | – |
| | 32 | 5.012 | 20.352 | – | – |
| SRAM | 16 | 6.644 | 0.685 | – | – |
| | 20 | 6.647 | 0.708 | – | – |
| | 24 | 6.650 | 0.731 | – | – |
| | 28 | 6.654 | 0.754 | – | – |
| | 32 | 6.658 | 0.778 | – | – |
| STTRAM | 16 | 3.444 | 4.803 | – | – |
| | 20 | 4.196 | 5.896 | – | – |
| | 24 | 4.970 | 6.984 | – | – |
| | 28 | 5.760 | 8.069 | – | – |
| | 32 | 6.559 | 9.152 | – | – |
| PCM | 16 | 1.449 | 3.242 | 1.621 | 1.621 |
| | 20 | 1.487 | 4.076 | 2.026 | 2.026 |
| | 24 | 1.526 | 4.891 | 2.431 | 2.431 |
| | 28 | 1.567 | 5.692 | 2.836 | 2.836 |
| | 32 | 1.612 | 6.483 | 3.242 | 3.241 |

Figure 5.3: Dynamic energy reduction - BasketballDrill

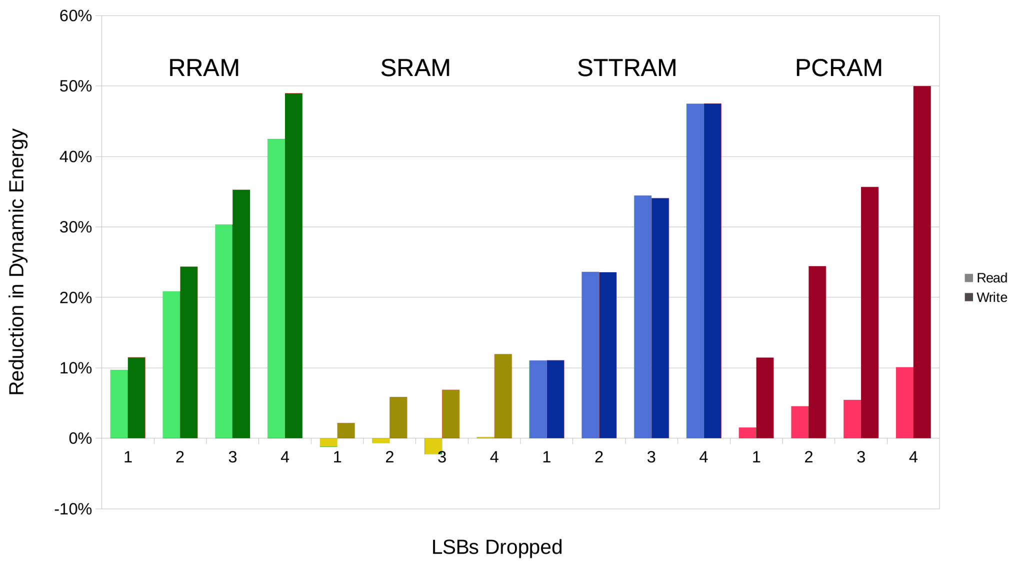


Figure 5.4: Dynamic energy reduction - Cactus

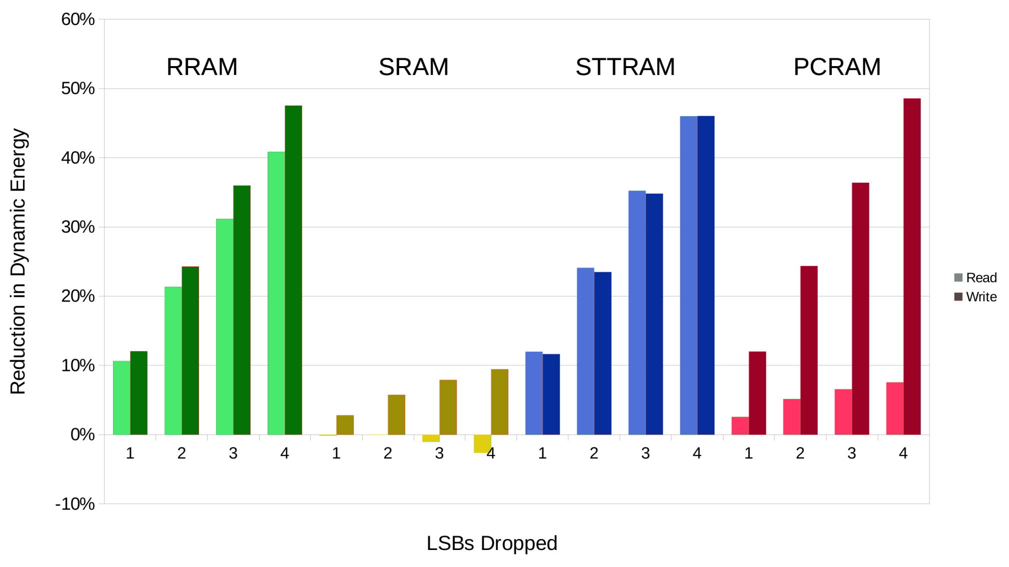
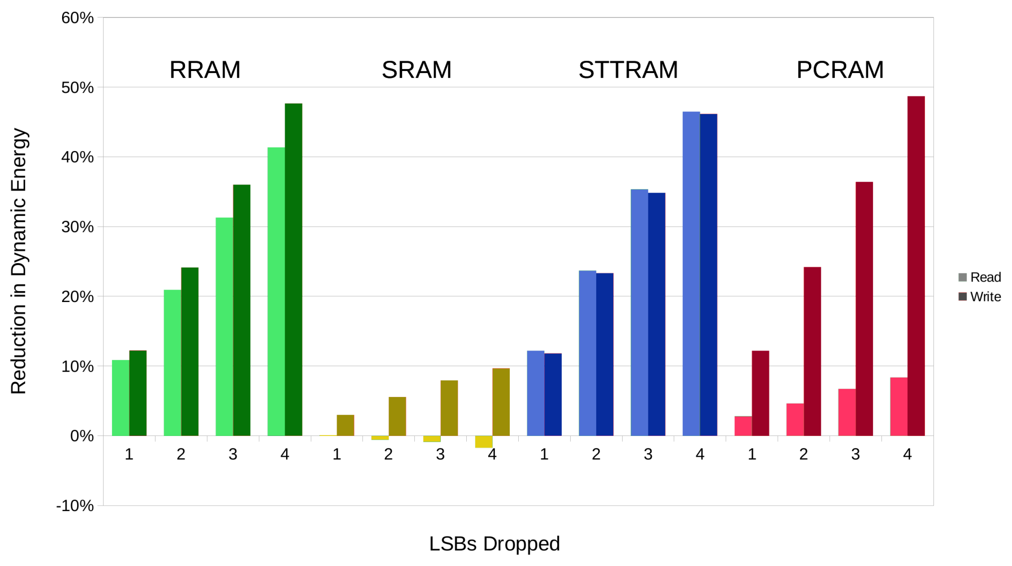


Figure 5.5: Dynamic energy reduction - Traffic



# 6 CONCLUSIONS

This work presented an analysis of approximate memory applied in motion estimation for video coding, using the bit dropping technique. The motivations for exploring video coding as an application for approximate memories were discussed, and the analysis of related works suggested that exploring NVMs in this context could result in even higher energy savings when compared to conventional CMOS SRAM caches.

Bit dropping was implemented in the x265 video encoding software, with the methodology implementation through a Python script, containing the different tools used to simulate the cache memory behavior. The cache access profiling was extracted using Cachegrind, and the results of simulations were parsed accordingly. The approximate storage was evaluated in the cache memory hierarchy level, where the impact in video compression and quality was measured using BD-Rate, a metric that combines both PSNR and bit rate metrics. The energy savings with bit dropping were simulated using NVSim, a non-volatile memory simulator, for SRAM, STT-RAM, RRAM, and PCMs.

The results have shown that the approximation impact in BD-Rate can be tolerable, especially considering the energy savings that can be achieved - up to 49% reduction in dynamic energy. The cache access behavior did not seem to be impacted by the approximations introduced with bit dropping. We conclude that NVMs are ideal for the bit dropping technique when evaluating the reduction that can be achieved in the dynamic energy.

This implementation of bit dropping is focused on the SAD modules used by the motion estimation step. Although it is the most relevant module in terms of computing demand, in order to better understand the impact of a real approximate memory chip for video coding reference frames, the impact in the Sum of Absolute Transformed Differences (SATD) and motion compensation modules should also be evaluated. This additional aspect of bit dropping will be explored in a future work.

Another possible follow up for this work is the addition of other approximate memory techniques to the same evaluation methodology. By doing that, we will have a more complete scenario to compare how different approximation schemes perform in the context of video coding applications.

## REFERENCES

Abreu, B. et al. Exploiting absolute arithmetic for power-efficient sum of absolute differences. In: **2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS)**. [S.l.: s.n.], 2017. p. 522–525.

ARJOMAND, M. et al. Boosting access parallelism to pcm-based main memory. In: **2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)**. [S.l.: s.n.], 2016. p. 695–706. ISSN 1063-6897.

Bjontegaard, G. **Calculation of average PSNR differences between RD-curves**. 2001. <https://www.itu.int/wftp3/av-arch/video-site/.../VCEG-M33.doc>. Accessed: 2019-06-14.

BOSSEN, F. **Common Test Conditions and Software Reference Configurations**. 2013. JCTVC-L1100.

CHANDRA, V.; SHARMA, P. **White Paper: Write Assist in Low-Voltage SRAMs**. [S.l.], 2012.

CHUN, K. C. et al. A scaling roadmap and performance evaluation of in-plane and perpendicular mtj based stt-mrams for high-density cache memory. **IEEE Journal of Solid-State Circuits**, v. 48, n. 2, p. 598–610, Feb 2013. ISSN 0018-9200.

CISCO. **Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper**. 2017.

Dong, X. et al. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 31, n. 7, p. 994–1007, July 2012. ISSN 0278-0070.

FRUSTACI, F. et al. Approximate srams with dynamic energy-quality management. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 24, n. 6, p. 2128–2141, June 2016. ISSN 1063-8210.

FRUSTACI, F. et al. 13.8 a 32kb sram for error-free and error-tolerant applications with dynamic energy-quality management in 28nm cmos. In: **2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)**. [S.l.: s.n.], 2014. p. 244–245. ISSN 0193-6530.

Gonzalez, R.; Gordon, B. M.; Horowitz, M. A. Supply and threshold voltage scaling for low power cmos. **IEEE Journal of Solid-State Circuits**, v. 32, n. 8, p. 1210–1216, Aug 1997. ISSN 0018-9200.

Grellert, M.; Bampi, S.; Zatt, B. Complexity-scalable hevc encoding. In: **2016 Picture Coding Symposium (PCS)**. [S.l.: s.n.], 2016. p. 1–5. ISSN 2472-7822.

HAO. **Design exploration of emerging nano-scale non-volatile memory**. New York, NY: Springer, 2014. ISBN 978-1-4939-0550-8.

HENNESSY, J. **Computer architecture : a quantitative approach**. Waltham, MA: Morgan Kaufmann, 2012. ISBN 978-0-12-383872-8.

IRANPOUR, A.; KUCHCINSKI, K. Memory Architecture Evaluation for Video Encoding on Enhanced Embedded Processors. In: **Proc. of Int. Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation**. [S.l.: s.n.], 2006. v. 4017, p. 309–320.

JACOB, B. **Memory systems : cache, DRAM, disk**. Burlington, MA: Morgan Kaufmann Publishers, 2008. ISBN 978-0-12-379751-3.

JUNG, M. et al. Invited: Approximate computing with partially unreliable dynamic random access memory — approximate dram. In: **2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2016. p. 1–4.

LEE, I. et al. Priority based error correction code (ecc) for the embedded sram memories in h.264 system. **Journal of Signal Processing Systems**, v. 73, n. 2, p. 123–136, Nov 2013. ISSN 1939-8115. Disponível em: <https://doi.org/10.1007/s11265-013-0736-4>.

MATIVI, A. **Exploration of Approximate Memory Architectures in HEVC**. 2018. Resumo no XXX Salão de Iniciação Científica UFRGS.

MATIVI, A.; MONTEIRO, E.; BAMPI, S. Memory Access Profiling for HEVC Encoders. In: **2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS)**. [S.l.: s.n.], 2016. p. 243–246.

Mittal, S.; Vetter, J. S.; Li, D. A survey of architectural approaches for managing embedded dram and non-volatile on-chip caches. **IEEE Transactions on Parallel and Distributed Systems**, v. 26, n. 6, p. 1524–1537, June 2015. ISSN 1045-9219.

MONTEIRO, E. et al. Rate-Distortion and Energy Performance of HEVC and H.264/AVC Encoders: A Comparative Analysis. In: **Proc. of IEEE Int. Symposium on Circuits and Systems**. [S.l.: s.n.], 2015. p. 1278–1281. Doi=10.1109/ISCAS.2015.7168874.

Monteiro, E. et al. Rate-distortion and energy performance of hevc video encoders. In: **2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)**. [S.l.: s.n.], 2014. p. 1–8.

MULTICOREWARE. **x265 HEVC Encoder / H.265 Video Codec**. 2019. <http://x265.org>. Accessed: 2019-06-14.

NETHERCOTE, N.; SEWARD, J. Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation. In: **Proceedings of the ACM SIGPLAN**. [S.l.: s.n.], 2007. v. 42, p. 89–100.

NGUYEN, D. T. et al. An approximate memory architecture for a reduction of refresh power consumption in deep learning applications. In: **2018 IEEE International Symposium on Circuits and Systems (ISCAS)**. [S.l.: s.n.], 2018. p. 1–5. ISSN 2379-447X.

OHM, J. R. et al. Comparison of the Coding Efficiency of Video Coding Standards-including High Efficiency Video Coding (HEVC). **IEEE Trans. on Circuits and Systems for Video Technology**, v. 22, p. 1669–1684, 2012.

RANJAN, A. et al. Approximate memory compression for energy-efficiency. In: **2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)**. [S.l.: s.n.], 2017. p. 1–6.

SAMPAIO, F. et al. dsvm: Energy-efficient distributed scratchpad video memory architecture for the next-generation high efficiency video coding. In: **2014 Design, Automation Test in Europe Conference Exhibition (DATE)**. [S.l.: s.n.], 2014. p. 1–6. ISSN 1530-1591.

SAMPAIO, F. et al. Approximation-aware multi-level cells stt-ram cache architecture. In: **2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)**. [S.l.: s.n.], 2015. p. 79–88.

SHAFIQUE, M. et al. Invited: Cross-layer approximate computing: From logic to architectures. In: **2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2016. p. 1–6.

SHAFIQUE, M.; HENKEL, J. Low Power Design of the next-generation High Efficiency Video Coding. In: **Proceedings of the Design Automation Conference**. [S.l.: s.n.], 2014. p. 274–281. Doi=10.1109/ASPDAC.2014.6742902.

Silveira, B. et al. Power-efficient sum of absolute differences hardware architecture using adder compressors for integer motion estimation design. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 64, n. 12, p. 3126–3137, Dec 2017. ISSN 1549-8328.

SINANGIL, M. et al. Memory Cost vs. Coding Efficiency Trade-Offs for HEVC Motion Estimation Engine. In: **Proc. of IEEE Int. Conference on Image Processing**. [S.l.: s.n.], 2012. p. 1533–1536. Doi=10.1109/ICIP.2012.6467164.

STAZI, G. et al. Introducing approximate memory support in linux kernel. In: **2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)**. [S.l.: s.n.], 2017. p. 97–100.

SULLIVAN, G. et al. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Trans. on Circuits and Systems for Video Technology**, v. 22, p. 1649–1668, 2012.

Suock Chung et al. Fully integrated 54nm stt-ram with the smallest bit cell dimension for high density memory application. In: **2010 International Electron Devices Meeting**. [S.l.: s.n.], 2010. p. 12.7.1–12.7.4. ISSN 2156-017X.

TEIMOORI, M. T. et al. Adam: Adaptive approximation management for the non-volatile memory hierarchies. In: **2018 Design, Automation Test in Europe Conference Exhibition (DATE)**. [S.l.: s.n.], 2018. p. 785–790. ISSN 1558-1101.

VANNE, J. et al. Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs. **IEEE Trans. on Circuits and Systems for Video Technology**, v. 22, 2012.

WIEGAND, T. et al. Overview of the H.264/AVC Video Coding Standard. **IEEE Trans. on Circuits and Systems for Video Technology**, v. 13, p. 560–576, 2003.

Xu, Q.; Mytkowicz, T.; Kim, N. S. Approximate computing: A survey. **IEEE Design Test**, v. 33, n. 1, p. 8–22, Feb 2016. ISSN 2168-2356.

ZAND, R. et al. Scalable adaptive spintronic reconfigurable logic using area-matched mtj design. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v. 63, n. 7, p. 678–682, July 2016. ISSN 1549-7747.

ZATT, B. et al. Run-time Adaptive Energy-Aware Motion and Disparity Estimation in Multiview Video Coding. In: **Proceedings of the Design Automation Conference**. [S.l.: s.n.], 2011. p. 1026–1031. Doi=10.1145/2024724.2024950.

ZEINALI, B. et al. Progressive scaled stt-ram for approximate computing in multimedia applications. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v. 65, n. 7, p. 938–942, July 2018. ISSN 1549-7747.

# Appendices

## Appendix A - TG1

# A Survey on Approximate Memory Technologies

**Ana Clara Mativi de Souza**[1]

`acmsouza@inf.ufrgs.br`

[1]Informatics Institute – Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brazil

***Abstract.*** *The Approximate Computing paradigm explores the energy, area and performance benefits that can be achieved through hardware level approximations for applications that are resilient enough to produce useful outputs even in the presence of errors, called Error-Tolerant Applications (ETAs).*
*Approximate Memories are explored in ETAs, with many implementations available in the literature, covering single or multiple memory hierarchy levels, using emerging and established memory technologies. A wide range of papers brings the challenge of comparing and analyzing the approaches taken. In this work, we attempt to categorize these techniques and find new ground for exploration.*

## 1. Introduction

Approximate Computing is a design paradigm that can be employed in situations where results do not have to be exact – applications that can tolerate a certain amount of approximation errors and still produce useful outputs. This leads to lower power consumption and overall performance gain in computer systems. Such techniques are explored in Error-Tolerant Applications (ETAs), which include image and video processing, neural networks, computer vision, web searches, machine learning, etc.

Media applications are intrinsically inexact, since the data that is encoded varies with the capture equipment used, such as the camera or sensors. Furthermore, slight changes in pixel values will not be perceived by the user due to human perceptual limitations.

For other applications, it may not be feasible to get the best result due to time limitations, yet the approximate result is acceptable, like in heuristics and probabilistic algorithms [Shafique et al. 2016].

Approximate Memories bring these approximations to the data storage level. By relaxing the the output quality requirements, gains in power/performance can be achieved, which draws interest since memory is the major bottleneck for performance and power [Sampaio et al. 2014].

The literature presents broad interpretations of the term ”Approximate Memory”. Paper [Sampaio et al. 2015] defines approximate storage as a memory unit without protection against error for read/write operations. Some papers, such as [Frustaci et al. 2016] employ techniques in circuit level to achieve approximate storage. Other works use software to simulate errors in memory (e.g. controlled bit-flip probability) such as [Stazi et al. 2017].

It is important to note that for most systems it is interesting to have exact storage for critical data, such as control variables, preventing errors that could cause application crashes and invalid outputs.

In Section 2 the literature overview is presented, which focuses on the different interpretations of Approximate Memory and the techniques that are employed to achieve that. Related works are discussed in Section 3. Finally, in Section 4 the idea for the continuation of this work is presented.

## 2. Background

### 2.1. Memory Hierarchy

The Memory Hierarchy organizes computer storage into a hierarchy based on access time. Each lower level usually has a larger capacity for storing data, but is also slower to access. Figure 1 presents a simplified view of this organization, highlighting the cache hierarchy which follows the same principle regarding access time and storage capacity.

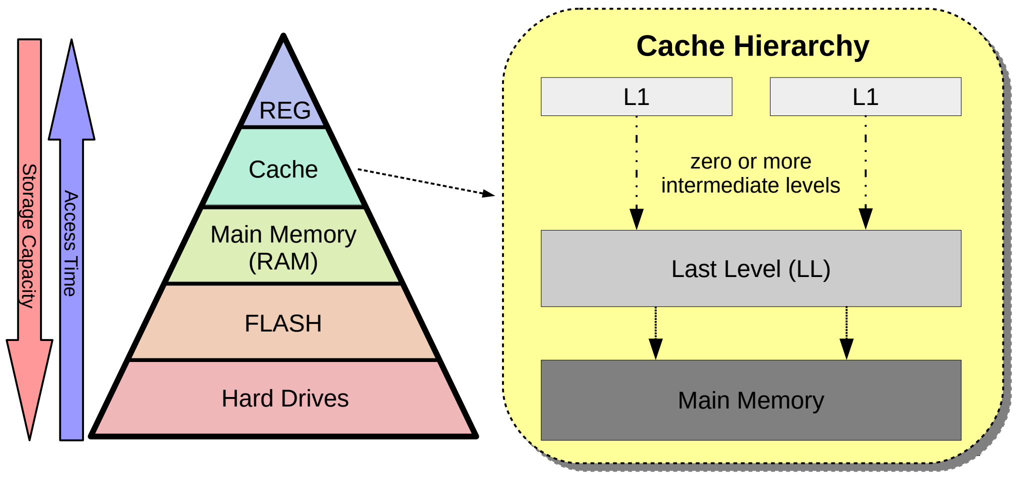


**Figure 1. Memory Hierarchy**

The Memory Hierarchy relies on the effective access of data to bridge the gap between the performance of processors and external memory. Considering how data is accessed, in different layers of the hierarchy, specially in the caches, spatial and temporal localities are explored. Temporal locality refers to the reuse of data shortly after it's first accessed. Spatial locality refers to data that is stored physically close (i.e. in the same word), since when the LL cache requests a data element from the Main Memory, a full cache line is read.

### 2.2. Random-Access Memory

Random-Access Memory (RAM) allows for the direct access of any word in memory using bitline (BL) and wordline (WL) to select the cells. Usually, multiple bits are accessed by the same address (depending on word size), and some RAM devices can also access multiple words in one cycle.

### 2.3. Memory Technologies

#### 2.3.1. Static RAM

Static Random-Access Memory (SRAM) stores bits in cells consisting of 2 inverters and 2 access transistors, for a total of 6 transistors. SRAM is more expensive to produce, but is much faster and requires less dynamic power than DRAM. Because of that, SRAMs are usually used for processor caches.

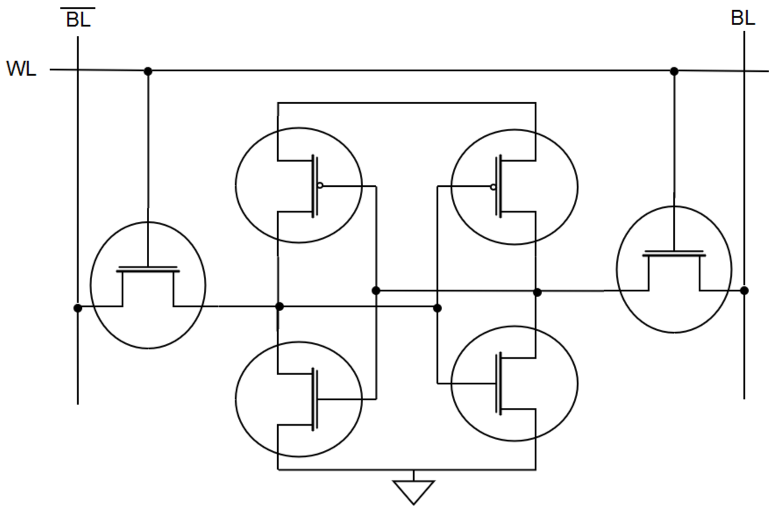


**Figure 2. 6 Transistor SRAM Cell**

Different SRAM cell schematics exist, such as 8 transistors (more robust, lower power consumption) and 4 transistors (higher static power consumption).

### 2.3.2. Dynamic RAM

Dynamic Random-Access Memory (DRAM) stores bits in cells that consist of a capacitor and a transistor. The capacitor can either be charged or discharged, representing bit states '0' or '1'.

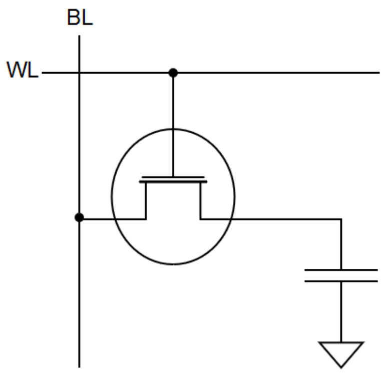


**Figure 3. DRAM Cell**

The capacitors holding data slowly discharge, so it's necessary to refresh the data stored in regular intervals to prevent the loss of information. The need to refresh the data in memory is the reason why it is called "dynamic", in contrast to SRAM which keeps the data as long as power is supplied. The simple cell circuitry allows DRAM to reach very high chip densities, making DRAM much cheaper per bit than SRAM.

### 2.3.3. Spin-Transfer Torque RAM

Spin-Transfer Torque Random-Access Memory (STT-RAM) uses a Magnetic Tunnel Junction (MTJ) device as the storage element in memory cells. The two possible states are defined according to the resistance (parallel/anti-parallel).

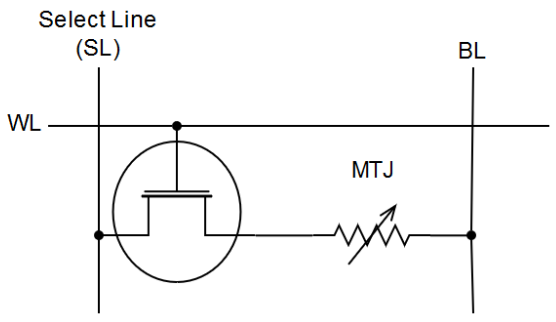


**Figure 4. STT-RAM Cell**

This memory technology is non volatile, presents low leakage and high speed access [Zand et al. 2016], which makes it a good candidate to substitute DRAMs. The drawbacks presented by STT-RAMs are the high write current, and the wrong write cycle (2x more than the necessary for SRAMs [Chun et al. 2013]). Finding ways to overcome these challenges has been a topic of intense research.

### 2.3.4. Phase Change Memory

Phase Change Memory (PCM) presents memory cells that switch between amorphous state (high resistance or RESET) and crystalline state (low resistance or SET) by applying an electrical current [Arjomand et al. 2016].

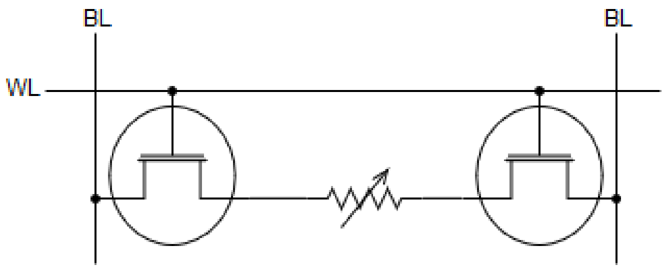


**Figure 5. PCM Cell**

In Multi-Level Cell Phase Change Memory (MLC-PCM) by dividing the resistance range it's possible to SET/RESET multiple times in order to represent multiple bits in a single cell. PCM presents multiple challenges that need to be addressed, such as the lifetime limitation due to the finite number of write operations a cell can endure. In MLC-PCM, a way to stretch the lifetime of the chip is to use approximations introduced by limiting the number of SET/RESET operations (thus representing different bits than intended). [Teimoori et al. 2018]

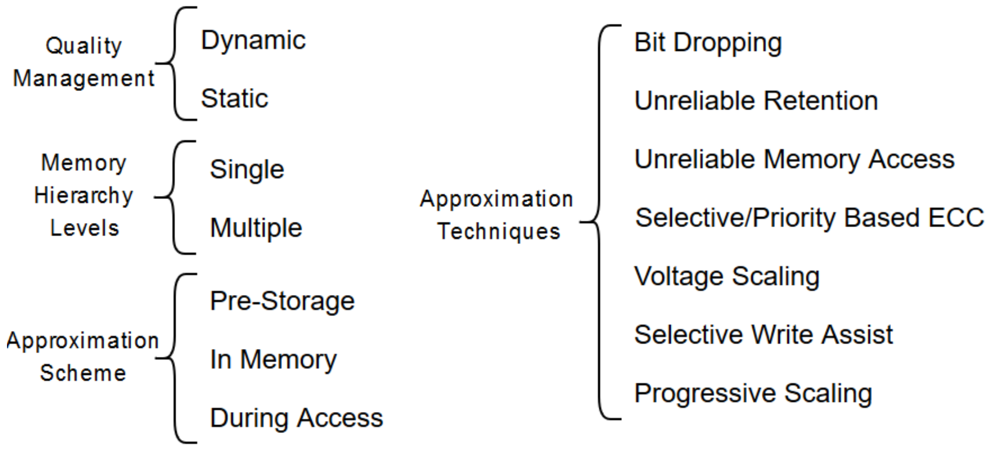


**Figure 6. Approximate Memory Overview**

## 2.4. Quality Management

Management schemes proposed in the literature are usually a static solution, (i.e. a certain gain is achieved for each solution), but some dynamic management proposals can also be found. Dynamic in this context means that it's possible to alter the quality of the output results while the application is running.

## 2.5. Memory Hierarchy Levels

Most papers focus on a single memory level, but it is also possible to explore approximate memory hierarchies by using multiple approximate memory levels; for example, with on-chip and off-chip memories. An exploration of this scenario is presented in [Teimoori et al. 2018].

## 2.6. Approximation Scheme

The approximation schemes refer to the moment when the approximation (loss of information) occurs. Three categories were identified:

- Pre-Storage: schemes that involve representing less bits per byte, lossy compression of data
- In Memory: schemes that may introduce errors while the data is in memory, such as Unrealiable Retention and Selective/Priority Based ECC
- During Access: schemes in which the errors may occur due to inaccuracy in read or write operations, which is the case for Unreliable Memory Access and Selective Write Assist

## 2.7. Approximation Techniques

### 2.7.1. Bit Dropping

Bit dropping refers to dropping one or more bits in each byte of the approximate memory portion. The usual approach is to drop bits from the Least Significant Bits (LSBs), since for images the Most Significant Bits (MSB) influence the Peak Signal-to-Noise Ration (PSNR) the most.

Paper [Frustaci et al. 2016] takes this approach one step further: since all the bit positions incur the same energy/access cost, boosting only the MSB bits during a write, while using the LSBs as Error-Correcting Codes (ECC) for the MSB improves the robustness of the data overall while still saving energy when compared to the exact approach.

### 2.7.2. Unreliable Retention

This technique focuses on DRAMs, and consists of refreshing with longer intervals than the maximum guaranteed retention of the cells, or ignoring resheshes altogether. With this, the bits that store a '1' may lose their stored value due to the discharge of the capacitors. It is important to note that if data rows are accessed, this effect will be avoided for a period of time – in a read operation, the bits will be amplified (similar to what would happen in a refresh), and in a write operation, the data is overwritten.

### 2.7.3. Unreliable Memory Access

This refers to approximations introduced by errors during write/read operations in memory. [Sampaio et al. 2015] uses this concept in MLC-STT-RAMs. [Stazi et al. 2017] implements the same idea at the Operating System level, allowing for the programmer to allocate approximate memory, which may introduce errors in bits with a controlled probability for each memory access.

### 2.7.4. Selective/Priority Based ECC

Selective Error-Correcting Codes (SECC) are ECCs applied to a portion of the memory cells (i.e. the bits that have more impact on quality). The other bits have no redundancy and are prone to errors [Frustaci et al. 2016]. A Priority Based ECC (PB-ECC) has also been proposed and validated for video coding in H.264, applying a stronger ECC for the MSBs than the LSBs [Lee et al. 2013].

### 2.7.5. Voltage Scaling

This technique consists of applying a lower supply voltage to SRAMs. Scaling the voltage down impacts PSNR drastically to achieve better power savings. To avoid this problem and still achieve considerable power savings, the literature suggests applying voltage scaling together with other techniques.

### 2.7.6. Selective Write Assist

The Write Assist (WA) techniques aid the bit cell in changing the state during write operations and are widely used in low power SRAMs [Chandra and Sharma 2012]. Some WA implementations available are wordline boosting, negative bitline, VDD lowering and VSS raising. Negative Bitline Boosting (NBB) is used alongside Voltage Scaling in

[Frustaci et al. 2014] for the MSBs only, reducing the cost added by the WA and increasing the robustness of the MSBs.

### 2.7.7. Progressive Scaling

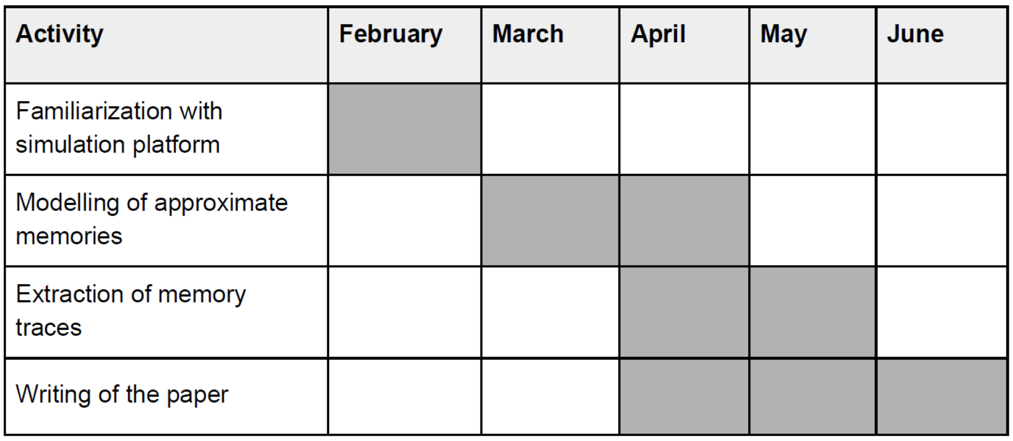
This technique uses STT-RAM cell scaling in order to provide lower BER to the MSB, while LSBs are stored in cells with higher BER, following the same idea used for bit dropping which relates the quality of the output with different bit positions.

## 3. Related Work

Specifically for video coding, many papers focus on approximate hardware accelerators. Recently, more papers have been published targeting approximate memories for video coding. Table 3 presents an overview of the literature according to the parameters explained in Section 2.

| Paper | Quality Management | MH Levels | Approx. Scheme | Techniques |
|---|---|---|---|---|
| [Frustaci et al. 2016] | Dynamic | Single | Pre/During Storage | BD, VS, SECC, SWA |
| [Sampaio et al. 2015] | Static | Single | During Access | Unreliable Access |
| [Stazi et al. 2017] | Static | Single | During Access | Unreliable Access |
| [Zeinali et al. 2018] | Static | Single | Pre-Storage | Progressive Scaling |
| [Ranjan et al. 2017] | Dynamic | Single | Pre-Storage | Unreliable Retention |
| [Nguyen et al. 2018] | Dynamic | Single | During Storage | Unreliable Retention |
| [Jung et al. 2016] | Static | Single | During Storage | Unreliable Retention |
| [Teimoori et al. 2018] | Dynamic | Multiple | During Access | Unreliable Access |

## 4. Work Proposal



| Activity | February | March | April | May | June |
|---|---|---|---|---|---|
| Familiarization with simulation platform | ▨ | | | | |
| Modelling of approximate memories | | ▨ | ▨ | | |
| Extraction of memory traces | | | ▨ | ▨ | |
| Writing of the paper | | | ▨ | ▨ | ▨ |

**Figure 7. Activities Timeline for 2019**

Video coding has been explored in Approximate Computing with interesting gains in performance/power in exchange for acceptable degradation in quality. The previous studies targeting cache memory hierarchy [Mativi et al. 2016] support further investigation related to memories, specially regarding to the modules that perform the most memory accesses. HEVC runs in platforms with complex memory hierarchies, and because of the great impact both in execution time and energy consumption associated to memory accesses, improving the memory architecture, video coding and memory configurations can bring interesting trade-offs in execution time, coding efficiency, and power savings.

This work proposes the use of Approximate Memories for High Efficiency Video Coding (HEVC or H.265), with the development of a methodology to evaluate the impact in the output quality and energy consumption. The memory access pattern of the HEVC encoder will be recorded as a memory trace and used to extract power results. The quality degradation produced by the approximations will be evaluated with objective metrics used by the literature such as PSNR, BD-BR.

The activities are organized for a 5 month timeline in Figure 7.

## References

Arjomand, M., Kandemir, M. T., Sivasubramaniam, A., and Das, C. R. (2016). Boosting access parallelism to pcm-based main memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 695–706.

Chandra, V. and Sharma, P. (2012). White paper: Write assist in low-voltage srams. Technical report, ARM.

Chun, K. C., Zhao, H., Harms, J. D., Kim, T., Wang, J., and Kim, C. H. (2013). A scaling roadmap and performance evaluation of in-plane and perpendicular mtj based stt-mrams for high-density cache memory. *IEEE Journal of Solid-State Circuits*, 48(2):598–610.

Frustaci, F., Blaauw, D., Sylvester, D., and Alioto, M. (2016). Approximate srams with dynamic energy-quality management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(6):2128–2141.

Frustaci, F., Khayatzadeh, M., Blaauw, D., Sylvester, D., and Alioto, M. (2014). 13.8 a 32kb sram for error-free and error-tolerant applications with dynamic energy-quality management in 28nm cmos. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 244–245.

Jung, M., Mathew, D. M., Weis, C., and Wehn, N. (2016). Invited: Approximate computing with partially unreliable dynamic random access memory — approximate dram. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–4.

Lee, I., Kwon, J., Park, J., and Park, J. (2013). Priority based error correction code (ecc) for the embedded sram memories in h.264 system. *Journal of Signal Processing Systems*, 73(2):123–136.

Mativi, A., Monteiro, E., and Bampi, S. (2016). Memory Access Profiling for HEVC Encoders. In *2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS)*, pages 243–246.

Nguyen, D. T., Kim, H., Lee, H., and Chang, I. (2018). An approximate memory architecture for a reduction of refresh power consumption in deep learning applications. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5.

Ranjan, A., Raha, A., Raghunathan, V., and Raghunathan, A. (2017). Approximate memory compression for energy-efficiency. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6.

Sampaio, F., Shafique, M., Zatt, B., Bampi, S., and Henkel, J. (2014). dsvm: Energy-efficient distributed scratchpad video memory architecture for the next-generation high efficiency video coding. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6.

Sampaio, F., Shafique, M., Zatt, B., Bampi, S., and Henkel, J. (2015). Approximation-aware multi-level cells stt-ram cache architecture. In *2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pages 79–88.

Shafique, M., Hafiz, R., Rehman, S., El-Harouni, W., and Henkel, J. (2016). Invited: Cross-layer approximate computing: From logic to architectures. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6.

Stazi, G., Menichelli, F., Mastrandrea, A., and Olivieri, M. (2017). Introducing approximate memory support in linux kernel. In *2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pages 97–100.

Teimoori, M. T., Hanif, M. A., Ejlali, A., and Shafique, M. (2018). Adam: Adaptive approximation management for the non-volatile memory hierarchies. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 785–790.

Zand, R., Roohi, A., Salehi, S., and DeMara, R. F. (2016). Scalable adaptive spintronic reconfigurable logic using area-matched mtj design. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(7):678–682.

Zeinali, B., Karsinos, D., and Moradi, F. (2018). Progressive scaled stt-ram for approximate computing in multimedia applications. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65(7):938–942.